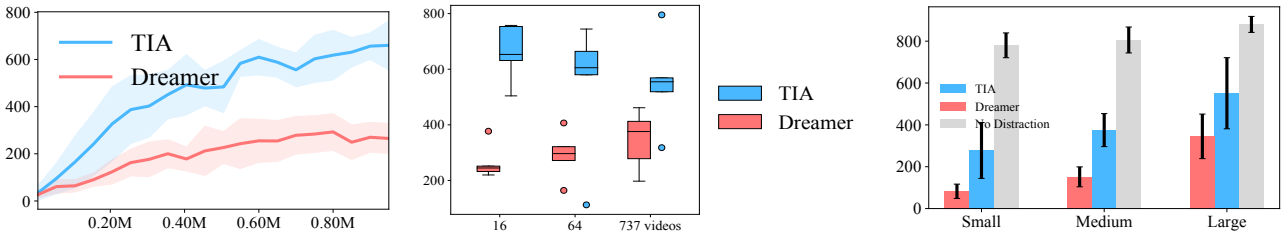
A. Experimental Details

A.1. Additional Ablation

The learning curves for the DMC domains in Figure 5 has large variance. We run a two-tailed test for performance at 1M steps, the results are: $[t=3.17,\ p=0.014]$, $[t=2.28,\ p=0.052]$, and $[t=2.21,\ p=0.061]$ on *Hopper*, *Walker* and *Cheetah*, respectively. A major source of variance in the result is that the set of 737 videos in the natural video dataset are only sampled 1000 times in total (at the beginning of each episode). Therefore the model is constantly surprised by visually distinct videos (see outliers in Figure 9b), leading to large variance in the episodic reward for all algorithms considered. To control for this, we present additional results on *Cheetah Run* that only uses 16 videos in Figure 9a and a comparison across $16, 64, 737$ videos in a box plot (Figure 9b). The variance decreases significantly when this is controlled for, showing that TIA outperforms Dreamer. One interesting observation is that Dreamer performs worse with a smaller number of videos (potentially because it prioritizes memorization of the video background).

We further reproduce Figure 1 while including TIA's performance in Figure 9c. TIA significantly outperforms Dreamer irrespective of the model size.



(a) With 16 videos (b) Variance from number of videos (c) Varying model size

Figure 9: (a) With 16 videos used in the environment, the learning curve has much smaller variance, showing TIA significantly outperforms Dreamer. (b) Performance at 1M steps over the five seeds, using different number of videos in the environment. (c) TIA outperforms Dreamer irrespective of the model size at 1M steps.

A.2. Performance Gap When Learning In The Presence of Visual Distractions

When complex visual distractors are present in the observations, state-of-the-art model-based agents struggle to maintain their original sample efficiency and asymptotic performance. We produce this gap in details with Figure 10, where we include seven additional (ten in total) domains from the DeepMind Control suite. In *Finger Spin* the gap almost disappears due to poor performance of the model-based agent in the clean environment – making the domain ill-suited for testing our proposal. Note that the maximum episodic return on these tasks is calibrated to 1000 (Tassa et al., 2018). We additionally provide the ratio for the number of pixels that are replaced by the background video for selected domains as a proxy for how much visual distraction is introduced: Walker 63%, Hopper 77%, Cheetah: 83%.

Our intent with this paper and the proposal, learning *Task Informed Abstractions*, is to close this gap such that model-based agents can retain their performance even when learning in the presence of complex visual distractions.

A.3. Bridging the Gap by Learning Task Informed Abstractions

We produce the complete result on DeepMind Control Suite, including the seven additional tasks from above. We set up the experiments by matching the total number of parameters so that each one of TIA's two models is only half in size as the single world model from Dreamer. This comparison is a disadvantage to our method because the smaller task model runs the risk of being too small to capture the set of task-relevant features in its entirety.

Despite of this disadvantage in model capacity, TIA is able to reduce the gap on *Walker Run*, *Hopper Hop*, and *Reacher Easy* while making the gap significantly smaller on *Hopper Stand* and *Cheetah Run* (see Figure 10). On *Walker Stand* and *Walker Walk*, the gap is small to begin with. Therefore our method cannot bring much benefit. Domains such as *Cartpole Swingup* and *Cup Catch* are devastatingly hard for Dreamer to master under visual distraction due to the partial-observable/small object manipulation nature of the domains. To conclude, since our implementation of TIA is a Dreamer with an additional capability of ignoring task-irrelevant features, it would not perform worse than Dreamer. However, tasks that are inherently hard for model-based methods would remain hard for TIA.

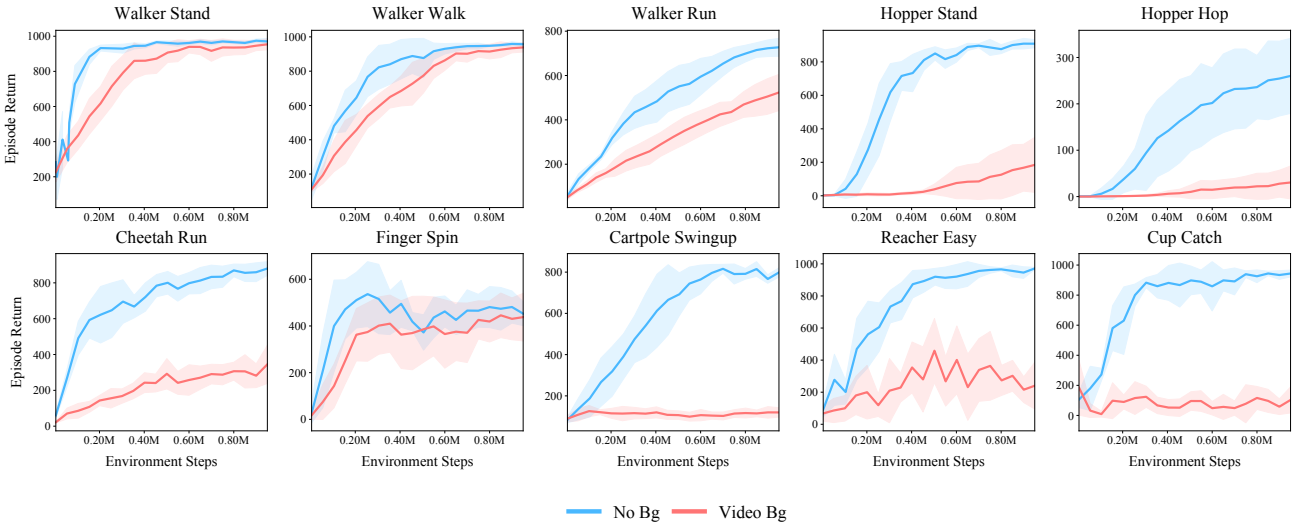


Figure 10: **The data efficiency and performance gap** when learning with video distraction backgrounds.

A.4. Understanding Failure Cases

In Section 4.5, we provide the principled approach to tuning the hyperparameters λ_{Radv} and λ_{Os}, where we balance these two terms to avoid either one of the two models taking over the entire reconstruction. We label these two extreme cases, where either the distractor model or the task model takes over, as type I and type II. We provide detailed renderings of these failure cases below Figure 12 in comparison to a successfully learned world model.

The first column in Figure 12a shows a successfully trained agent whose task model is able to perfectly reconstruct the walker agent. In type I failure mode (Figure 12b), the distractor model would take over the entire reconstruction, causing the task model to lose its grasp on the task-relevant features. This would prevent the policy from learning any useful behavior, which makes the collected reward practically random. With random rewards, the term $\mathcal{J}_{\text{Radv}}$ is ineffective at dissociating task-relevant features, causing training to fail. The policy performance under this scenario is usually close to that of a random policy. When this happens, we want to increase λ_{Radv} to dissociate the distractor model from the reward, or decrease λ_{Os} that weighs the distractor reconstruction term.

In the type II failure case (see Figure 12c), the task model takes over the reconstruction. In this case, the model degenerates into Dreamer without separating out the task-irrelevant features, and the performance is close to a dreamer agent with a smaller model. In this case, we would increase the weight λ_{Os} on the distractor reconstruction to encourage it to learn more features.

By tuning these two parameters λ_{Radv} and λ_{Os}, we were able to avoid these two failure modes and stabilize training on a wide variety of domains. A future direction would be to tune these to parameters automatically using the signals mentioned in Section 4.5.

B. Model Architecture Details

Model Detals and Sizes For fair comparisons, on DMC and ManyWorld we match the total number of parameters of our two world models combined with that of a single, large Dreamer model. The total number of parameters is 10 million on DMC and 1 million on ManyWorld. We divide the two models equally in size, with an additional image reconstruction head for the distractor model. *Dreamer-Inverse* has the same size for all model components as the Dreamer model, except it replaces the deconvolution head with an action prediction head for learning the inverse dynamics. We scale the model size by changing the width of each layer in the networks without changing the overall architecture or the depth of the networks. All other hyperparameters such as learning rates are kept the same as (Hafner et al., 2020).

On the Arcade Learning Environments, we compare against the state-of-the-art on this domain, DreamerV2 (Hafner et al., 2021), which uses a world model of 20 million trainable parameters. We made the task model the same size as the original

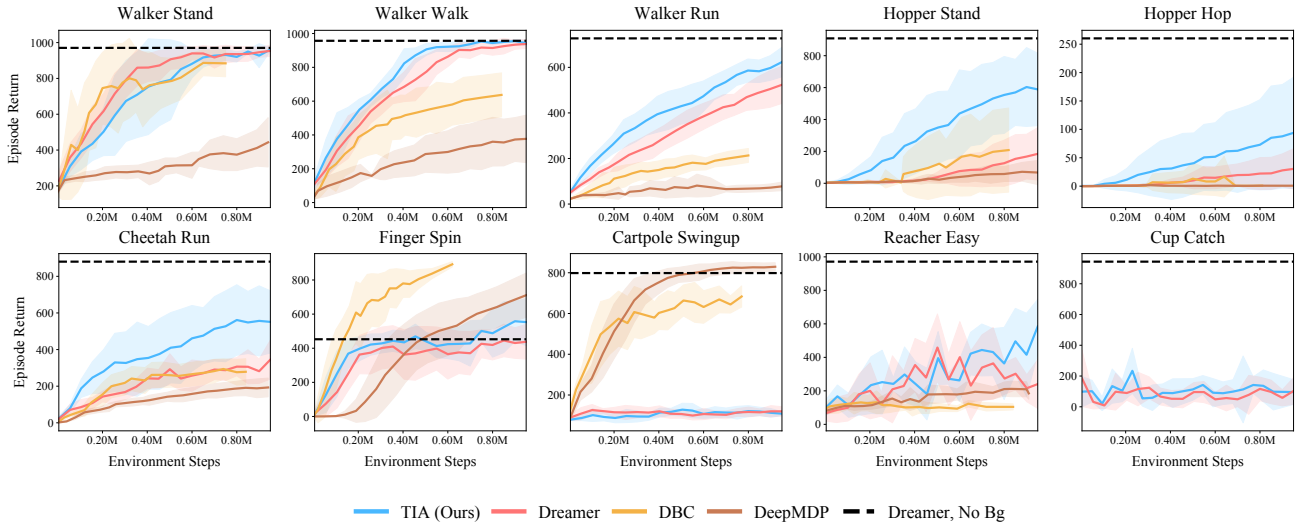


Figure 11: **Performance of Task Informed Abstractions** on ten DeepMind control domains. Zhang et al. 2021 did not evaluate DBC/DeepMDP on *Cup Catch* so we only report results of TIA and Dreamer for *Cup Catch*.

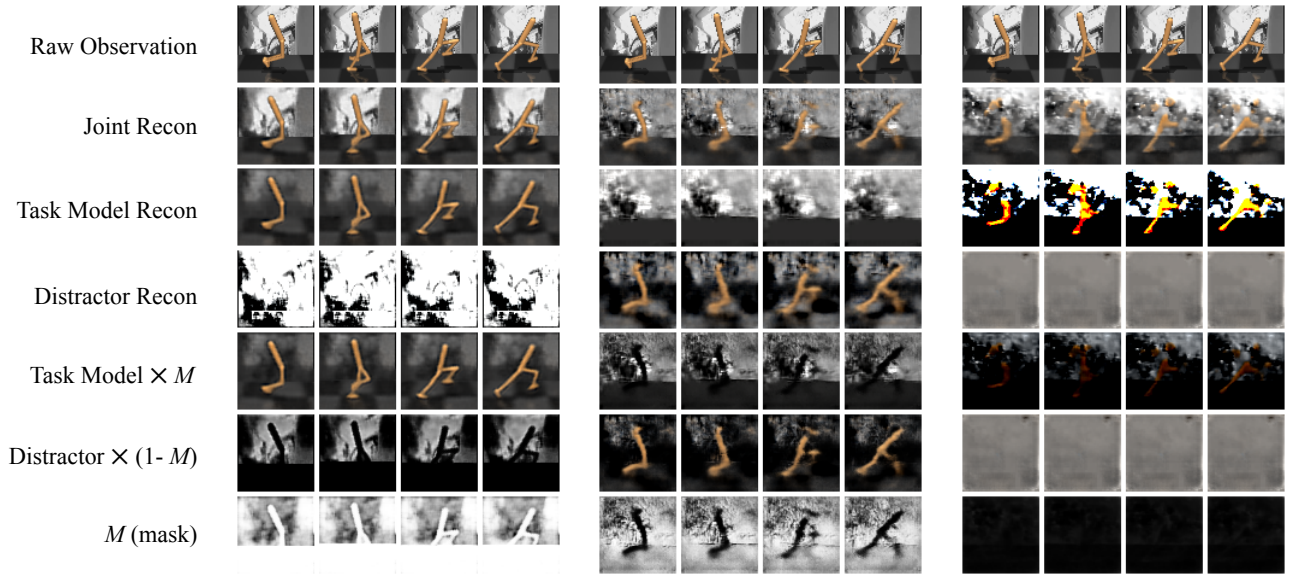
implementation while adding a smaller distractor model which contains 12 million parameters. A key difference between Dreamer and DreamerV2 is that the latter has an additional prediction head for the discount factor γ_t besides the standard reward prediction head. This discount factor head plays an instrumental role in allowing DreamerV2 to solve Atari games. Therefore we additionally dissociate the distractor model from information about the discount factor by adding an additional adversarial prediction loss. We use the same scale for the discount factor γ as that for reward: $\lambda_{\gamma\text{adv}} = \lambda_{\text{Radv}}$. All other hyperparameters such as learning rates are kept the same as (Hafner et al., 2021).

We use an input size of $32\times32\times3$ in ManyWorld, $64\times64\times3$ in DeepMind Control Suite, and $64\times64\times1$ in Atari games. We use grayscale for the natural video backgrounds, the same as previous work (Zhang et al., 2021).

Hyperparameters For fair comparisons, we did not tweak existing hyperparameters from Dreamer and used identical settings as Hafner et al. (2020) and Hafner et al. (2021). Our reward-dissociation scheme introduces two new hyperparameters λ_{Radv} and λ_{Os}. We scale the reward dissociation loss via λ_{Radv} such that the term matches reconstruction losses in magnitude. For this reason, the differences in scale in Table 2 mostly reflect the differences in input image sizes. We tweaked λ_{Os} to stabilize training. Detailed settings for each domain are in Table 2.

Table 2: Hyperparameters

Domain and Task	λ_{Radv}	λ_{Os}
ManyWorld, 1 Distractor	600.0	2.0
ManyWorld, 2 Distractor	150.0	2.0
Cartpole Swingup	30k	2.0
Hopper Stand	30k	2.0
Hopper Hop	30k	2.0
Cheetah Run	20k	1.5
Walker Run	20k	0.25
Walker Walk	20k	0.25
Walker Stand	20k	0.25
Finger Spin	30k	2.5
Reacher Easy	20k	1.0
All ATARI games	2k	1.0



(a) Success

(b) Failure Type 1: The distractor model takes over in reconstruction

(c) Failure Type 2: The task model takes over in reconstruction

Figure 12: Detailed rendering from one successfully trained walker agent, and two failure cases of type 1 and type 2. In type 1 failure case, the distractor model takes over the entire reconstruction, rendering the task model ineffective for policy learning (random policy). In the type 2 failure case, the task model attempts to capture all factors of variation by itself, thus failing to perfectly reconstruct the image, also leading to sub-par policy performance in the lower 400.

C. Transfer to Novel Distractions

The task informed abstraction we introduce in this paper improves learning when distractions are present. To adapt to out-of-distribution scenarios unseen during training, additional architectural changes that reject distracting image features on the fly may be required. To provide a baseline and intuitions for this future direction, we evaluate how well existing agents perform under this type of domain shift. In Table 3 we take agents that are trained (1) without video background, (2) with background videos from the *driving car* class or (3) with white noise backgrounds, and evaluate against background videos from a different class, *walking the dog* (labeled as *transfer*, see Table 3).

Table 3: **DeepMind Control Transfer Performance** transfer to the video class *walking the dog* as background

Training Condition		Drmr, No Bg	Drmr, Video	TIA, Video	Drmr, Noise	TIA, Noise
Hopper Stand	In-domain	906.8 ± 29.3	183.8 ± 162.1	596.4 ± 234.1	769.7 ± 205.4	744.8 ± 75.8
	Transfer	18.1 ± 12.1	186.2 ± 142.2	629.4 ± 231.5	357.4 ± 206.7	354.9 ± 136.9
Walker Run	In-domain	728.2 ± 37.8	520.2 ± 84.4	625.3 ± 64.7	737.6 ± 26.7	696.9 ± 43.9
	Transfer	127.4 ± 34.0	530.9 ± 76.9	645.3 ± 78.4	341.1 ± 108.9	345.5 ± 138.0
Cheetah Run	In-domain	876.3 ± 36.0	325.7 ± 96.6	556.6 ± 167.7	754.9 ± 67.0	734.2 ± 163.4
	Transfer	21.1 ± 7.7	312.6 ± 115.7	557.4 ± 194.6	227.0 ± 75.1	309.5 ± 233.2

The Dreamer agent trained with no background distraction fails to transfer its performance when background videos are introduced at test time, which is expected. In the second experiment, we train both Dreamer and TIA with video background but test using videos from a different category. Dreamer did not learn as well as TIA as indicated by its poor performance in the training environments, but both methods retain their training performance post-transfer, unaffected by the change in the background video. As a control, we also train both methods using white noise as the background. The training and transfer performance are both identical between the two methods, and the transfer performance is worse than performance on white noise.

We additionally evaluate ManyWorld agents that are trained with (1) no distractor, (2) one distractor, or (3) two distractors, with an additional distractor (three distractors, see Table 4).

Table 4: **ManyWorld Transfer Performance** transfer to three distraction blocks.

Training Condition		Drmr, 0	Drmr, 1	Drmr, 2	TIA, 1	TIA, 2
ManyWorld	In-domain	246.0 ± 3.5	242.9 ± 5.4	217.4 ± 29.3	246.1 ± 1.7	245.8 ± 1.8
	Transfer	192.6 ± 18.9	198.4 ± 27.4	192.0 ± 35.1	185.7 ± 21.4	233.4 ± 6.5

Both results show that while TIA learns better from cluttered scenes, a mechanism to reject unseen backgrounds at decision time is required to transfer successfully. This points to the incorporation of attention as a potential avenue for future work.